The Complexity of Computing the Random Priority Allocation Matrix

Daniela Saban*and Jay Sethuraman†

January 2014; revised August 2014

Abstract

The Random Priority (RP) mechanism is a popular way to allocate n objects to n agents with strict ordinal preferences over the objects. In the RP mechanism, an ordering over the agents is selected uniformly at random; the first agent is then allocated his most-preferred object, the second agent is allocated his most-preferred object among the remaining ones, and so on. The outcome of the mechanism is a bi-stochastic matrix in which entry (i, a) represents the probability that agent i is given object a. It is shown that the problem of computing the RP allocation matrix is #P-complete. Furthermore, it is NP-complete to decide if a given agent i receives a given object a with positive probability under the RP mechanism, whereas it is possible to decide in polynomial time whether or not agent i receives object a with probability 1. The implications of these results for approximating the RP allocation matrix as well as on finding constrained Pareto optimal matchings are discussed.

1 Introduction

We consider the problem of allocating n objects to n agents, with each agent interested in consuming at most one unit across all objects. Agents have *strict* ordinal preferences over the objects. Perhaps the most common allocation mechanism for this problem is the *priority* mechanism (also called the *serial dictatorship* (SD) mechanism): in such a mechanism, there is a fixed ordering of the agents and the agents are invited to choose objects in that order. Thus, the agent who appears first in this ordering will pick his most-preferred object; the one appearing second will pick his most-preferred object among the ones that remain, etc. The priority mechanism is Pareto efficient, neutral (invariant to relabeling of the objects), non-bossy (no agent can alter some other agent's allocation without altering his own), strategy-proof, even group strategy-proof, and easy to compute. Its one major drawback, however, is that it fails *anonymity*—two agents with identical preferences and identical claims on the objects are not

*Graduate School of Business, Columbia University, New York, NY; dhs2131@columbia.edu

†IEOR Department, Columbia University, New York, NY; jay@ieor.columbia.edu. Research supported by NSF grant CMMI-0916453 and CMMI-1201045.

treated equally by the mechanism because one of them will appear before the other in the fixed ordering. A standard way to overcome this in a moneyless market is to *randomize* the initial ordering of the agents, yielding the *Random Priority* (RP) mechanism. In the RP mechanism, an ordering of the agents is chosen uniformly at random (each of the $n!$ orderings being equally likely), and the priority mechanism applied to the chosen ordering determines the outcome. Of course, finding the outcome is an easy task once the ordering is selected.

An alternative way to think about the RP mechanism is in terms of the *probabilistic* allocation that the agents receive under this mechanism—this can be expressed as a doubly-stochastic matrix X with x_{ia} representing (i) the probability that agent i receives object a (if the objects are indivisible); or (ii) the fraction of object a allocated to agent i (if the objects are divisible). The RP mechanism has been extensively analyzed in the literature for the allocation of both divisible and indivisible objects [5, 12], yet the computational complexity of finding the RP allocation matrix X is not fully understood. Our main result is that determining X exactly or even approximately is *difficult* in a sense that can be made precise using the theory of computational complexity.

In computational complexity theory, a *decision problem* is a question with a yes or no answer, depending on the values of some input parameters. As an example, the problem "Given a preference profile P, does agent i get object a in the priority mechanism with respect to some ordering σ of the agents?" is a decision problem. Indeed, we refer to this problem as the SD Feasibility problem. Complexity theory is concerned with understanding the computational resources needed to solve a problem, and to categorize problems into various *complexity classes* depending on how "easy" or "difficult" it is to find a solution. Two complexity classes play an important role in this paper: the class NP, defined as the set of decision problems having efficiently verifiable solutions and the class #P, introduced by [16] and defined as the counting version of the class NP. A decision problem is in NP if every "yes" instance can be efficiently verified using a polynomial-size certificate. As an example, consider the SD Feasibility problem stated earlier. If the problem has n agents, and if indeed there is an ordering σ of the agents such that the SD mechanism with respect to σ gives object a to agent i, this can be easily verified in polynomial-time by running the SD mechanism with the ordering σ. The ordering σ serves as the "certificate" in the definition of NP. Two additional notions are needed in what follows: A problem is *hard* for a given complexity class if it is as difficult as the most difficult problems in the class. When a problem is both a member of the class and hard for that class, it is said to be *complete* for that class.

By definition, the problem of determining each entry x_{ia} of the RP allocation matrix X is equivalent to that of counting the number of orderings under which i obtains a when the SD mechanism is used. Therefore, this problem is in the class #P. We show that computing the RP allocation is indeed #P-complete, and thus suspected to be very difficult.[1] Independently of our work, Aziz et al. [3] proved that computing the RP allocation is #P-complete using a

[1]Toda's theorem implies that, for any problem in the polynomial hierarchy, there is a deterministic polynomial-time Turing reduction to a problem in #P, and therefore one call to a #P oracle suffices to solve any problem in the polynomial hierarchy in deterministic polynomial time [15].

different reduction.

Even though the problem of finding the RP allocation matrix for a given instance is not directly a decision problem, it can be solved as a sequence of decision problems of the following form: "Given an integer K, a preference profile P, an agent i and an object a, are there at least K orderings under which i is assigned a when the SD mechanism is used?". Our #P-completeness result already implies that this decision problem is NP-complete if K is part of the input. Nevertheless, the decision problem might be easy to solve for some fixed values of K. Towards that end, we study the computational complexity of the decision problems associated with the two extreme values of K: deciding whether an agent has probability exactly one of getting an object ($K = n!$), and deciding whether he has probability greater than zero ($K = 1$). We provide a polynomial-time algorithm to solve the former and show that the latter is NP-complete. We further show that the problem of deciding whether an agent has a positive probability of obtaining an object is equivalent to deciding whether there is a Pareto efficient matching in which a subset of objects must be matched. Computing matchings with constraints has been a topic of interest to the research community, as it naturally arises in social choice applications in which some type of affirmative action is imposed [1, 6].

In spite of the prominence of the RP mechanism, there is surprisingly little prior work regarding its computational complexity. As already mentioned, Aziz et al. [3] independently establish the #P-completeness of computing the RP allocation matrix using a different reduction. Their work, however, does not address the complexity of approximating the RP allocation. As a corollary of our NP-completeness result, we establish that the RP allocation matrix is even hard to approximate. Similarly, to the best of our knowledge, only the work of Kavitha and Nasre [11] considers the complexity of finding constrained optimal matchings, but they use the notion of popularity to define an optimal matching. Instead, we use the notion of Pareto efficiency, which is more standard and widely used in the literature.

The rest of this paper is organized as follows. In Section 2, we formally state the problem and provide some useful definitions and notation. In Section 3, we show that computing the RP allocation is #P-complete. In Section 4, we discuss the complexity of two decision problems associated with the RP mechanism. We provide a polynomial-time algorithm for deciding whether an agent has probability exactly one of getting an object. In addition, we show that deciding whether an agent has probability greater than zero of getting an object is NP-complete and discuss the implications of this result. We end with some suggestions for further research in Section 5.

2 Preliminaries

An instance of the house allocation problem is a tuple $\mathcal{I} = (\mathcal{A}, \mathcal{O}, P)$ consisting of a set of agents $\mathcal{A}$, a set of objects $\mathcal{O}$ and a preference profile $P = (P_1, \ldots, P_{|\mathcal{A}|})$, where each P_i is a *strict* ordering of the set of objects $\mathcal{O}$. The ordering P_i represents agent i's preferences over the objects: given two objects a and b, we say that i prefers a to b if a appears before b in the ordering P_i. We write $a >_{P_i} b$ or simply $a >_i b$ if i prefers a to b. Our preference model assumes

that each agent finds every object acceptable, although all the results hold more generally with some obvious modifications. We also assume that $|\mathcal{A}| = |\mathcal{O}|$ unless otherwise noted. We will sometimes refer to an instance by just P, as often the sets of agents and objects are clear from context and are implicit in P anyway.

A *matching* is a bijective function from the set of agents to the set of objects. Given a matching μ, we say that $\mu(i) = a$ if agent i receives object a in the matching μ. A matching μ is *Pareto-efficient* if there is no other matching ν such that $\nu(i) \geq_i \mu(i)$ for all agents $i \in \mathcal{A}$, with at least one inequality strict. That is, a matching is Pareto efficient if no agent can be better off without requiring another agent to be worse off. A (Pareto-efficient) *deterministic mechanism* is a function that assigns a (Pareto efficient) matching to every preference profile P. A (Pareto efficient) *randomized mechanism* is a function that maps each preference profile P to a distribution over (Pareto efficient) matchings.

Let Σ be the set of all orderings of $\mathcal{A}$. For $\sigma \in \Sigma$, let $\sigma(k)$ be the k^{th} agent according to order σ and $\sigma^{-1}(i)$ be the position of agent i in σ. Given a preference profile P and an ordering $\sigma \in \Sigma$, the *serial dictatorship* (SD) mechanism (also known as priority mechanism) works as follows: agent $\sigma(1)$ is assigned his most-preferred object, then agent $\sigma(2)$ is assigned his most-preferred object among the remaining ones, and so on. The matching found by the SD mechanism on a preference profile P and ordering σ is denoted $SD(P, \sigma)$, or simply μ_σ when the preference profile P is clear.

The *random priority* (RP) mechanism (also called random serial dictatorship) selects an ordering from Σ uniformly at random and then finds the outcome $SD(P, \sigma)$, where σ is the selected ordering. The outcome of the RP mechanism is a bi-stochastic allocation matrix X: the rows are indexed by agents and the columns by objects, with the entry x_{ia} indicating the probability that agent i will receive object a in the RP mechanism (in the case of indivisible objects), or the fraction of a that i receives (in the case of divisible objects). To explicitly indicate the dependence of the RP allocation matrix on the preference profile, we denote as $X(P)$ or $RP(P)$, and its $(i, a)^{th}$ entry by $X(P, i, a)$ or $RP(P, i, a)$. It should be clear from the definition of the RP mechanism that $X(P, i, a) = RP(P, i, a) = |\{\sigma \in \Sigma : \ \mu_\sigma(i) = a\}|/n!$.

3 The complexity of Random Priority

As mentioned in Section 2, the definition of RP implies $RP(P, i, a) = |\{\sigma \in \Sigma : \ \mu_\sigma(i) = a\}|/n!$. Therefore, determining an entry (i, a) of the RP allocation matrix is equivalent to counting the number of orderings under which i gets a. We refer to this problem as the SD Count problem, which is formally defined as follows:

SD Count

Input. A strict preference profile P, associated with a set of agents $\mathcal{A}$ and a set of objects $\mathcal{O}$, an agent i and an object a

Output. The number of orderings $\sigma \in \Sigma$ in which $\mu_\sigma(i) = a$.

We now examine the complexity of the SD Count problem. Recall that the SD Feasibility problem (as defined in the introduction) is a problem in NP, and, as SD Count is the counting version of SD Feasibility, it is a problem in the class #P . We will show that SD Count is indeed #P-complete. To do so, we introduce the Linear Extension Count problem. A *partially ordered set* (or *poset*) is a set Q equipped with an irreflexive and transitive relation $<_Q$. A *linear extension* of a poset Q on n elements is a linear ordering $\prec$ of the elements such that $x \prec y$ whenever $x <_Q y$. The linear extension count problem is defined as follows:

Linear Extension Count
Input. A partially ordered set Q.
Output. The number $N(Q)$ of linear extensions of Q.

Brightwell and Winkler [4] proved that Linear Extension Count is #P-complete. We now show that Linear Extension Count can be reduced to SD Count, therefore establishing the hardness of the latter.

Theorem 1. SD Count *is #P-complete .*

Proof. Clearly, given an ordering $\sigma \in \Sigma$, one can verify in polynomial time whether $\mu_\sigma(i) = a$, and so SD Count is in #P . To show that SD Count is #P-complete, we reduce Linear Extension Count to SD Count. Given a poset Q, consider the following instance of SD Count with

$$\mathcal{A} = \{i \ : \ i \in Q\} \cup \{F\}, \text{ and, } \mathcal{O} = \{o_i \ : \ i \in Q\} \cup \{o_F\}.$$

In words, we have one agent and one object for each element of the poset Q, a special agent F and a special object o_F. Each agent $i \neq F$ ranks o_j ahead of o_i if and only if $j <_Q i$ in the poset Q; he then ranks the object o_i followed by the special object o_F; the remaining objects appear after o_F in any arbitrary order. Thus, the preferences for agent i will look as follows:

$$P_i \ = \ \underbrace{\{o_j \ : \ j <_Q i\}}_{\text{in any arbitrary order}} \ , \ o_i, \ o_F, \ \underbrace{\{o_j \ : \ j \neq i, F, j \not<_Q i\}}_{\text{in any arbitrary order}}.$$

Finally, the special agent F ranks the special object o_F last, and has an arbitrary preference ordering over the remaining objects.

Let σ be a fixed ordering of agents. Recall that $\sigma(k)$ is the k^{th} agent according to order σ and $\sigma^{-1}(i)$ is the position of agent i in σ. We prove the result by showing that $\mu_\sigma(F) = o_F$ if and only if $\sigma^{-1}(F) = n + 1$ and $\{\sigma(1), \ldots, \sigma(n)\}$ is a linear extension of Q. As a consequence, being able to determine the probability that the special agent F gets the special object o_F under the RP mechanism will imply an ability to compute the number of linear extensions of the given poset Q. Because the latter problem is #P-complete, so is the former.

Suppose σ is such that $\mu_\sigma(F) = o_F$. As o_F is agent F's last choice, it follows that (i) $\sigma^{-1}(F) = n + 1$; and (ii) every other agent received an object that they preferred to o_F. Therefore, for each $i \neq F$, $\mu_\sigma(i) \in \{o_j \ : \ j <_Q i\} \cup \{o_i\}$. We claim that, in fact, $\mu_\sigma(i) = o_i$ for all i and we prove the result by induction on n. The claim is clearly true for all the *minimal*

elements of Q: if k is such an element, then the preference ordering corresponding to agent k in the SD Count instance has o_k as the first element and o_F as the second; as each such k appears before the special agent F and does not receive the object o_F, it must be the case that each such k receives object o_k. Thus removing all the minimal elements from Q and their corresponding objects from $\mathcal{O}$ does not change the assignment for the rest of the agents. This also implies that for any pair of elements $i, j \in Q$ with $i <_Q j$, agent i appears before agent j in the ordering σ: for otherwise, j appears before i, and object o_i is still available when it is j's turn to choose an object, so $\mu_\sigma(j)$ should be at least as good as o_i according to agent j, contradicting the fact that $\mu_\sigma(j) = o_j$. This establishes that σ restricted to the first n positions is a linear extension of Q.

For the converse, suppose that $\sigma^{-1}(F) = n+1$ and that $\{\sigma(1), \ldots, \sigma(n)\}$ is a linear extension of Q. Then agent $\sigma(1)$ must correspond to a minimal element of the poset and must be assigned object $o_{\sigma(1)}$ as that is his most-preferred object. Removing this agent and object from the problem, we get a smaller instance with the same properties, and the result follows by induction, once we observe that the result is trivially true for $n = 1$. □

4 Decision problems associated with Random Priority

Given the preference profile P involving n agents and n objects, an integer $1 \leq k \leq n!$, an agent i and an object a, one can ask whether the number of orderings of the agents for which the SD mechanism gives a to i is *at least* k. The #P-completeness of computing the RP allocation matrix proves that this problem is NP-complete: for otherwise, we can determine the exact value of $RP(P, i, a)$ by doing a binary search over k. This would involve solving $\log(n!) = \Theta(n \log n)$ instances of this problem, each with a different value of k. Here we consider the same problem, but for *fixed* k; specifically, the two natural "extremal" values of k—that of $k = n!$ and $k = 1$. We address the following two questions: for a given preference profile, (i) does agent i have a positive probability of getting object a (SD Feasibility)?; and (ii) does agent i *always* get object a (SD Unique Assignment)?

4.1 The SD Feasibility problem

We now turn our attention to the SD Feasibility problem, which is formally defined as follows:

SD Feasibility
Input. A preference profile P, an agent i and an object a.
Output. Is there an ordering σ such that i obtains a in $SD(P, \sigma)$?

We show the somewhat surprising result that SD Feasibility is NP-complete by constructing a reduction from the problem of finding a minimum-cardinality maximal matching in a subdivision graph.

A *matching* in a graph $G = (V, E)$ is a subset M of edges such that no two edges in M share a vertex. The size of a matching M is the number of edges in M. A *maximal matching*

is a matching M with the property that $M \cup \{e\}$ is not a matching for any edge $e \in E \setminus M$. A *minimum-cardinality maximal matching*, or simply, *minimum maximal matching* is a maximal matching of minimum size. The decision version of the minimum maximal matching problem can be stated as follows:

MINIMUM MAXIMAL MATCHING
Input. A graph $G = (V, E)$ and an integer K.
Output. Is there a maximal matching in G of size at most K?

Let $G = (V, E)$ be a given graph. The subdivision graph of G is obtained by splitting each edge $e \in E$, and by locating a new vertex in the middle. Formally, it is the bipartite graph $S(G)$ with vertex set $V' = V \cup E$, and edge set

$$E' = \{\{e, v\} \mid e \in E, v \in V, \text{ and } v \text{ is incident with } e \text{ in } G\}.$$

It is known that MINIMUM MAXIMAL MATCHING is NP-complete on subdivision graphs [9].

Theorem 2. SD FEASIBILITY *is NP-complete.*

Proof. Given σ, one can verify in polynomial time if i gets a under $SD(P, \sigma)$ and therefore the problem is in NP. To show that this problem is NP-complete, we reduce MINIMUM MAXIMAL MATCHING on subdivision graphs to SD FEASIBILITY. Let $G' = (V' := V \cup E, E')$ –a subdivision graph of $G = (V, E)$– and K –an integer– be an instance of MINIMUM MAXIMAL MATCHING. Suppose $V = \{v_1, \ldots, v_n\}$, $E = \{e_1, \ldots, e_m\}$. Each $e_i \in E$ connects two different vertices v_{p_i} and v_{q_i} with $p_i < q_i$. Note that the subdivision graph has edges (e_i, v_{p_i}) and (e_i, v_{q_i}) for each $e_i \in E$. Without loss of generality, we may assume $m \geq n$.

We construct an instance of SD FEASIBILITY as follows:

- There are $3m + 1$ agents—two agents for each $e_i \in E$ and $m + 1$ special agents. The two agents corresponding to each e_i are labeled e_i^1 and e_i^2; m special agents are denoted $F_1, F_2, \ldots, F_m$ and the remaining special agent is denoted D.

- There are $3m+1$ objects—m corresponding to the elements of E and labeled $o_1, o_2, \ldots, o_m$; n corresponding to the elements of V and labeled $v_1, v_2, \ldots, v_n$; $m+1$ special objects labeled $o_{F_1}, \ldots, o_{F_{m+1}}$, and finally $m - n$ additional (dummy) objects, denoted $d_1, \ldots, d_{m-n}$, to enforce the constraint $|\mathcal{A}| = |\mathcal{O}|$. Thus the set of objects $\mathcal{O} = \{o_1, \ldots, o_m\} \cup \{v_1, \ldots, v_n\} \cup \{o_{F_1}, \ldots, o_{F_{m+1}}\} \cup \{d_1, \ldots, d_{m-n}\}$.

The preferences are defined as follows:

- $P(e_i^1) = \ o_i,\ v_{p_i},\ v_{q_i},\ o_{F_{m+1}},\ o_{F_m}, \ldots,\ o_{F_1}.$
- $P(e_i^2) = \ o_i,\ v_{q_i},\ v_{p_i},\ o_{F_{m+1}},\ o_{F_m}, \ldots,\ o_{F_1}.$
- $P(F_i) = \ o_{F_1},\ o_{F_2}, \ldots,\ o_{F_{m+1}}.$

- $P(D) = \ o_{F_{m+1}},\ o_{F_m}, \ldots,\ o_{F_1}$.

In any preference list, the objects not shown can be appended to that list arbitrarily. To give some intuition behind the preference structure: agents $F_1, \ldots, F_n$ rank all the special objects before any other object and rank them in *ascending* index order. The two edge agents corresponding to e_i rank *their* edge object o_i first, followed by their vertex objects, but ordered differently: the first copy ranks v_{p_i} before v_{q_i} whereas the second copy does the opposite; this is then followed by all the special objects, but arranged in *decreasing* index order. Finally, the special agent D ranks all the special objects in *decreasing* index order first. The ranking of the other objects in the preference lists is not important. Two points about the preference structure deserve mention: first, note that the F agents rank the special objects in *ascending* index order, whereas all other agents rank these objects in *descending* index order. Second, the special agents F could be omitted from the problem if we are allowed to have more objects than agents in the reduction, and as such they are introduced only to maintain balance between the number of agents and number of objects; as we shall see in a moment, there is a lot of freedom in how these agents are treated in the reduction.

We claim that there is a maximal matching $M \subseteq E'$ of G' such that $|M| \leq K$ if and only if there exists an ordering σ such that agent D obtains $o_{F_{K+1}}$ in $SD(P, \sigma)$.

Given a maximal matching M of G' with $|M| = \ell \leq K$, we construct an ordering σ of the agents such that D obtains $o_{F_{K+1}}$ in $SD(P, \sigma)$. Observe that in the graph G', the vertex e_i is connected to exactly two vertices v_{p_i} and v_{q_i}, so *at most* one of these two edges can be in M.

- If $(e_i, v_{p_i}) \in M$, rank agent e_i^2 ahead of e_i^1; if $(e_i, v_{q_i}) \in M$, rank agent e_i^1 ahead of e_i^2. If M has ℓ edges, this step will determine 2ℓ agents, and these agents are ranked ahead of all the other agents; but we are free to order these 2ℓ agents any way we like as long as we respect the relative ranking of the pair of agents corresponding to a fixed edge e_i as just mentioned.

- From the remaining $(m - \ell)$ edges in G' that are unmatched in M, select a subset S of exactly $m - K$ edges (note that this is possible as $\ell \leq K$). For each $e_i \in S$, rank agent e_i^1 before e_i^2.

- Rank agent D.

- Complete the ordering by adding the remaining edge agents and the agents F_j, $1 \leq j \leq m$, in an arbitrary order.

We now show that agent D will receive the object $o_{F_{K+1}}$ in the ordering just constructed.

If e_i is matched, then one of its copies will be assigned o_i and the other copy will be assigned v_{p_i} or v_{q_i}, depending on whether e_i was matched to v_{q_i} or v_{p_i}. In any case, if e_i is matched, both e_i^1 and e_i^2 will receive one of their first two choices.

Suppose e_i is unmatched, and suppose $e_i \in S$. Then, agent e_i^1 will get o_i and so agent e_i^2 cannot be assigned o_i; moreover, by the maximality of M, it must be that *both* v_{p_i} and v_{q_i}

are matched in M (otherwise one could add one of the edges involving e_i to M), and so the objects v_{p_i} and v_{q_i} are already assigned to a higher priority agent in our ordering. Thus, agent e_i^2 must be assigned a special object o_{F_j} for some j. Since $|S| = m - K$, $o_{F_{m+1}}, o_{F_m}, \ldots, o_{F_{K+2}}$ will be taken by the agents $\{e_i^2 : \ e_i \in S\}$. The next agent in the ordering is agent D, who according to his preferences will get $o_{F_{K+1}}$. Thus, given a maximal matching of size at most K, the constructed ordering σ is such that agent D obtains $o_{F_{K+1}}$ in $SD(P, \sigma)$, which establishes the "only if" part of the claim.

Now suppose that there exists an ordering σ such that agent D obtains $o_{F_{K+1}}$ in $SD(P, \sigma)$. We argue that there is an ordering in which all agents of type e appear before any special agent of type F, and such that D still receives $o_{F_{K+1}}$. To that end, suppose D is the l^{th} agent in σ. First, note that at most K agents of type F can be before D in σ. Furthermore, if we consider an ordering obtained from σ by removing all agents of type F that appear before D and placing them after D, the allocation of all agents of type e before position l will remain the same by the structure of the preferences. Therefore, we may assume that σ only contains agents of type e_i^j before position l. In addition, for every σ' that differs from σ in the ordering after position l, the allocation of the first l agents (including D) will not change. Hence, we may assume that all agents of type e_i^j appear before agents of type F_j in σ.

Effectively, we have established that if there is an ordering in which D receives object $o_{F_{K+1}}$, then there is an ordering in which all the "edge" agents appear before any special agent of type F, and such that D still receives $o_{F_{K+1}}$. In the rest of the proof, we assume that the given ordering is of this type.

Let $M = \{(e_i, v_j) : \ e_i^1$ or e_i^2 obtain v_j under $\sigma\}$. We argue that M must be a maximal matching of G. First, we show that M is a matching. Because all agents of type e appear before any agent of type F in σ, exactly one of $\{e_i^1, e_i^2\}$—the one that appears earlier—will obtain o_i for every $1 \leq i \leq m$. Hence, at most one of $\{e_i^1, e_i^2\}$ can be allocated an object of type v, implying that each e_i appears in at most one edge in M. On the other hand, by the definition of the serial dictatorship mechanism, each object of type v is allocated to at most one agent, and therefore it can appear in at most one edge in M. Hence, we conclude that M is indeed a matching. The maximality of M follows by the preference structure: if v_{p_i} is unmatched, then when it was the turn of the second copy of e_i to choose an object, v_{p_i} was not chosen; this can happen only if that copy of e_i chose v_{q_i}; in particular, e_i must be matched. A similar argument applies when v_{q_i} is unmatched. We conclude that M is a maximal matching of G. It remains to be shown that M has size at most K. Note that m agents of type e get their associated objects and at least $m - K$ get objects of type o_{F_j}. Then, at most K agents of type e will get an object of type v_j, and $|M| \leq K$ which completes the proof. □

4.1.1 Implications of the hardness of the SD Feasibility problem.

Theorem 2 has two strong implications. The first implication is related to the inapproximability of the RP mechanism. During the past decades, it has been shown that it is possible to design polynomial-time algorithms for *approximately* counting the number of solutions of some #P-

complete problems. Indeed, #P-complete problems admit only two possibilities: they either allow polynomial approximability to any required degree, or they cannot be approximated [14]. The former possibility is captured in the definition of a *fully polynomial randomized approximation scheme* (FPRAS). Formally, consider a problem whose counting version f is #P-complete. A randomized algorithm A is an FPRAS for this problem if, for each instance x and error parameter $\epsilon > 0$, $Pr[|A(x) - f(x)| \leq \epsilon f(x)] \geq 3/4$, and the running time of A is polynomial in $|x|$ and $1/\epsilon$. If the decision version of a counting problem is NP-complete, the counting problem itself cannot admit an FPRAS unless NP = RP, which is the complexity class consisting of problems that can be solved in randomized polynomial time[10].[2] Therefore, we have the following corollary:

Corollary 1. *The RP mechanism cannot admit an FPRAS unless NP = RP.*

Although the RP allocation matrix cannot be efficiently approximated, it is possible to distinguish efficiently (with high probability) the entries of the RP allocation matrix with high values from those with low values. Given preference profile P, an agent i and an object a, suppose we sample r orderings independently and uniformly at random and, for each ordering σ_j with $1 \leq j \leq r$, we set $X_j = 1$ if $SD(i, a, \sigma_j) = 1$ and $X_j = 0$ otherwise. Note that $Pr[X_j = 1] = RP(i, a)$. Let $X = \sum_{j=1}^{r} X_j$, and let $\overline{RP}_r(i, a) = X/r$ be our estimate for the real value of $RP(i, a)$ when using a sample of size r. One question that naturally arises is how large does r need to be in order to be able to distinguish, with high probability, if a certain entry $RP(i, a) = 0$ or is it bigger than a certain $q > 0$.

We can now use the Hoeffding's inequality [8] to obtain the following bound:

$$\Pr\left(|\overline{RP}_r(i, a) - RP(i, a)| \geq \delta\right) \leq 2\exp\left(-2r^2\delta^2\right)$$

This means that the probability that the estimate deviates more than δ from the real value of the RP entry is exponentially small in r and δ.

A second implication of Theorem 2 is related to the complexity of computing Pareto efficient matchings under constraints. We first highlight a different way of thinking about the SD Feasibility problem: Let $U(i, a)$ be the set of objects agent i prefers over a according to P_i. We say that M is a *partial Pareto efficient matching* if every agent in M prefers the object he is matched to over all objects unmatched in M, and no trade involving a subset S of agents in M can make all agents in S better off. Note that the unmatched agents might find unmatched objects admissible, and therefore M may not be a Pareto efficient matching.

We claim that there exists an ordering under which agent i is allocated object a if and only if there is a partial Pareto efficient matching in which all the objects in $U(i, a)$ are matched and a is unmatched. To prove the "if" part, suppose such a matching exists and denote it by M. Then, there exists an ordering σ_M involving the $|M|$ agents matched in M under which the RP mechanism will give matching M as an output. We can now extend σ_M to an ordering σ over all agents, by defining $\sigma(j) = \sigma_M(j)$ for all $1 \leq j \leq M$, $\sigma(|M| + 1) = i$ and adding the remaining

[2]Similarly to the problem P = NP, the problem of whether NP = RP is open and it suspected to be false.

agents to σ in any arbitrary ordering. It is easy to verify that agent i obtains object a under σ. To show the converse, let σ be an ordering under which i is allocated a. Note that the matching obtained by running the RP mechanism using ordering σ until position $\sigma^{-1}(i)$ must be a partial Pareto efficient matching in which all the objects in $U(i,a)$ are matched and a is unmatched.

We conclude by noting that finding a Pareto efficient matching in which all the objects in $U(i,a)$ are matched and a is unmatched is equivalent to finding a (partial) Pareto efficient matching in the *reduced* instance $\mathcal{I}_{\setminus\{i\}}^{\setminus\{a\}} = (\mathcal{A}\setminus\{i\}, \mathcal{O}\setminus\{a\}, P\setminus\{a\})$ with the constraint that all objects in $U(i,a)$ be matched.[3]

Based on this idea, we define the Constrained Pareto Efficient Matching problem as follows:

Constrained Pareto Efficient Matching
Input. A preference profile P in which agents may have inadmissible objects, a subset of objects Q.
Output. Is there a Pareto efficient matching in which all objects in Q are matched?

An immediate corollary of Theorem 2 is the following:

Corollary 2. Constrained Pareto Efficient Matching *is NP-complete .*

In a recent paper, Haeringer and Iehlé [7] study stability in two-sided matchings when the preferences for only one side of the market are known. In a two-sided matching model, each side of the market has preferences over the other side. For consistency with the existing literature, we refer to the sides of the market as men and women respectively. A matching is said to be *stable* if every matched agent finds his match acceptable, and if there is no pair of agents who would prefer to be matched to each other rather than to their current match (if any). In the model analyzed in [7], only the preferences of the women are known. While we do not know the preferences of the men, we do know that a man m finds a women w acceptable if and only if w ranks m somewhere in her preference ordering. Their goal is to say whether a pair of agents can be matched at a stable matching for *some* preference profile. Haeringer and Iehlé de signed a dynamic-programming algorithm for this problem with an exponential running time in the size of the input, but left open the possibility of a polynomial-time algorithm to solve this decision problem. Here we show that their problem is closely related to the problem of finding a constrained Pareto efficient matching in a one-sided matching problem, and so is NP-complete: the one-sided allocation problem with strict (but possibly incomplete) preferences is obtained by viewing the women as agents and men as objects. As before, let $U(w,m)$ be the set of men that w strictly prefers to m.

Claim 1. *In the above setting, a woman w and a men m can be matched at a stable matching for some preference profile if and only if there is a Pareto efficient matching for the women in which all the men in $U(w,m)$ are matched (in the problem where m and w are omitted).*

[3]Here, $P\setminus\{a\}$ represents the preferences P truncated so that every agent only lists as admissible those objects that he strictly prefers to a.

Proof. Suppose there is a Pareto efficient matching M in which all the men in $U(w,m)$ are matched. In this matching, clearly no unmatched woman can have an acceptable unmatched man. Suppose each matched man ranks his partner in M first; m ranks w first; and the preferences of the unmatched men are arbitrary. The resulting matching is stable: every matched man is married to his top-ranked woman; and none of the other men are acceptable to any available woman. This verifies the "if" part of the Claim.

To show the converse, suppose m and w are matched in some stable matching M. As M is stable, there are no blocking pairs. This means that all the men in $U(w,m)$ are matched, and that every woman must prefer her own match over any unmatched man. Thus the unmatched men in M will play no further role. If M is not Pareto efficient (in the problem in which m and w are omitted), consider the following reallocation of the matched pairs: there is a node for each woman; and there is an arc (w', w'') if and only if w'' is matched to the most-preferred remaining partner of w'; any cycles that form are cleared (so that the women involved in the cycle effect a Pareto improving swap), and the procedure is recursively applied.[4] Throughout this procedure, the set of matched men does not change; in particular, every man in $U(w,m)$ remains matched; and the final outcome is a Pareto efficient matching. □

4.2 The SD Unique Assignment problem

We now study the SD Unique Assignment problem, which is defined as follows:

> SD Unique Assignment
> *Input.* A preference profile P, an agent i and an object a.
> *Output.* Is it true that for every ordering $\sigma \in \Sigma$, agent i obtains a in $SD(P,\sigma)$?

Given an instance $\mathcal{I} = (\mathcal{A}, \mathcal{O}, P)$ and an object $a \in \mathcal{O}$, we define the reduced instance $\mathcal{I}^{\setminus\{a\}}_{\setminus\{i\}} = (\mathcal{A}\setminus\{i\}, \mathcal{O}\setminus\{a\}, P\setminus\{a\})$, where $P\setminus\{a\}$ represents the preferences P truncated so that every agent only lists as admissible those objects that he strictly prefers to a.

We start with the following lemma.

Lemma 1. *Given an agent i and an object a, $\mu_\sigma(i) = a$ for all $\sigma \in \Sigma$ if and only if:*

(1) a is agent i's top-choice.

(2) In every ordering σ such that $\sigma(i) = n$, $\mu_\sigma(j) >_j a$ for all $j \neq i \in \mathcal{A}$ (that is, in every ordering in which i is the last agent, all other agents get an object they like better than a).

Proof. Suppose $\mu_\sigma(i) = a$ for all $\sigma \in \Sigma$. Condition (1) must be trivially satisfied, as otherwise i will not choose a whenever $\sigma^{-1}(i) = 1$. Furthermore, consider an ordering σ such that $\sigma^{-1}(i) = n$. Since a is not assigned to any of the first $n-1$ agents, it follows that all of them must get objects they prefer to a. The converse is even simpler: if conditions (1) and (2) are satisfied, object a would be available when it is agent i's turn to choose, and so i will be assigned a. □

[4] The reallocation mechanism we just described is the well-known Top-Trading Cycles (TTC) mechanism proposed by Shapley and Scarf [13].

Lemma 1 forms the basis of Algorithm 1, which solves the SD Unique Assignment problem by verifying both conditions. Condition (1) can be easily checked. To verify condition (2), note that every ordering $\sigma \in \Sigma$ induces a Pareto efficient matching and every efficient matching can be implemented with (at least) one ordering σ. Hence, condition (2) fails to hold if and only if there is a Pareto efficient matching in the *reduced* problem $\mathcal{I}^{\setminus\{a\}}_{\setminus\{i\}}$ of size at most $n-2$. In that case, at least one object and one agent of $\mathcal{I}^{\setminus\{a\}}_{\setminus\{i\}}$ must remain unmatched. The key idea is to first identify those objects that are candidates to remain unmatched in a Pareto efficient matching for the reduced problem, and solve a matching problem for each object in turn to find whether there exists a Pareto efficient matching in which they remain unmatched.

Algorithm 1 SD Unique Assignment

Input: An instance $\mathcal{I} = (\mathcal{A}, \mathcal{O}, P)$, and agent $i \in \mathcal{A}$ and an object $a \in \mathcal{O}$

Output: Is it true that, for every ordering $\sigma \in \Sigma$, agent i obtains a in $SD(P, \sigma)$?

If object a is not agent i's top choice, return FALSE.
Consider the reduced instance $\mathcal{I}' = (\mathcal{A}', \mathcal{O}', P') = \mathcal{I}^{\setminus\{a\}}_{\setminus\{i\}}$.
Let $\mathcal{S} = \{o \in \mathcal{O}' : \ o \notin P'_j \text{ for some } j \in \mathcal{A}'\}$.
For each $o \in \mathcal{S}$:

Let $A(o) = \{j \in \mathcal{A}' : \ o \in P'_j\}$ (set of agents that find o admissible).

Consider the bipartite graph $G(o) = ((\mathcal{A}', \mathcal{O}' \setminus \{o\}), E)$, where $(k, j) \in E$ if and only if agent k finds object j admissible in $\mathcal{I}'$ and likes j better than o.

Find a maximum matching M in $G(o)$, with the constraint that every vertex in $A(o)$ must be matched.

If such a matching exists, return FALSE.

Return TRUE.

Theorem 3. *Algorithm 1 solves the* SD Unique Assignment *problem in polynomial time.*

Proof. Clearly, Algorithm 1 runs in polynomial time. To show the correctness of the algorithm, we may assume that object a is agent i's most-preferred object, as otherwise i will not always be assigned a. Consider the reduced instance $\mathcal{I}' = (\mathcal{A}', \mathcal{O}', P') = \mathcal{I}^{\setminus\{a\}}_{\setminus\{i\}}$. Since every ordering $\sigma \in \Sigma$ induces a Pareto efficient matching and every efficient matching can be implemented with (at least) one ordering σ, condition (2) fails to hold if and only if we are able to find an efficient matching in the reduced problem $\mathcal{I}'$ of size at most $n-2$. Note that, in that case, at least one object and one agent of $\mathcal{I}'$ must remain unmatched. Let o be an unmatched object in an efficient matching M. Clearly, o must be inadmissible for at least one agent in $\mathcal{I}'$ (in particular, it must be inadmissible for all unmatched agents in M). Therefore, we will first identify those objects that are candidates to remain unmatched in a Pareto efficient matching for the reduced problem, and then we will solve a matching problem for each object in turn to find whether there exists a Pareto efficient matching in which they remain unmatched.

Let $\mathcal{S} = \{o \in \mathcal{O}' : \ o \notin P'_j \text{ for some } j \in \mathcal{A}'\}$, that is, the objects in $\mathcal{S}$ are those that are inadmissible for at least one agent and thus are candidates for being unmatched in *some* efficient matching of $\mathcal{I}'$. For each object $o \in \mathcal{S}$, let $A(o) = \{j \in \mathcal{A}' : \ o \in P'_j\}$ be the set of agents that find object o admissible. Whenever o is unmatched, all agents in $A(o)$ must be matched to an object they like better than o. For each $o \in \mathcal{S}$, we can either find a Pareto efficient matching in which o is unmatched or we can show that no such matching exists as follows: Consider a bipartite graph $G(o) = ((\mathcal{A}', \mathcal{O}'\backslash\{o\}), E)$, where $(k, j) \in E$ if and only if agent k finds object j admissible in $\mathcal{I}'$ and likes j better than o. Find a maximum matching M in $G(o)$, with the constraint that every vertex in $A(o)$ must be matched. If no such matching exists, then o must be matched in every Pareto efficient matching of $\mathcal{I}$. Otherwise, note that M contains at most $n-2$ edges as $|\mathcal{O}\backslash\{o\}| = n-2$, but it might not be efficient. This matching, however, can be transformed into an efficient matching by performing a set of Pareto improvements, as described in [2]. Nevertheless, no Pareto improvement can involve o as all agents that find o admissible were assigned better objects than o and the rest do not find o admissible. Hence, we were able to find a Pareto efficient matching of size at most $n-2$ and thus show that i does not always get a. □

5 Discussion

Due to its simplicity and compelling properties, the RP mechanism is one of the most popular mechanisms for allocating objects. The hardness results in this paper imply that any mechanism that relies on the knowledge of the RP allocation matrix is likely to be impractical when the number of objects is large. This is the case, for instance, if one uses the RP mechanism to allocate divisible objects, assuming agents still have unit demand.

We have shown that the RP allocation is not only hard to compute in general, but also hard to approximate. However, in some cases in which the preference domain is restricted, the RP allocation can be easy to compute. One example is the work by [5], who consider a scheduling problem involving unit-length jobs and deadlines, which could be different for different jobs. For this special case, the RP allocation can be computed efficiently. A natural question of interest is to determine precisely the conditions under which one can compute the RP allocation in polynomial time, or to identify other natural problems where such a result is possible.

Acknowledgements

We thank Rocco Servedio and Xi Chen for their comments on the paper and for a useful discussion about approximation algorithms with an additive error. We thank Guillaume Haeringer and Vincent Iehlé for telling us about their recent work and for posing the problem of identifying stable pairs in a two-sided matching problem where the preferences of the agents are known only on one side of the market. We are grateful to three anonymous referees for their thoughtful comments and suggestions on an earlier version of this paper.

References

[1] A. Abdulkadiroglu. College admissions with affirmative action. *International Journal of Game Theory*, 33:525 – 549, 2005.

[2] D. J. Abraham, K. Cechlarova, D. F. Manlove, and K. Mehlhorn. Pareto optimality in house allocation problems. In *Algorithms and Computation*, volume 3341 of *Lecture Notes in Computer Science*, pages 3–15. 2005.

[3] H. Aziz, F. Brandt, and M. Brill. The computational complexity of random serial dictatorship. *Economics Letters*, 121(3):341 – 345, 2013.

[4] G. Brightwell and P. Winkler. Counting linear extensions is#p-complete. DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, 1990.

[5] H. Crés and H. Moulin. Scheduling with opting out: Improving upon random priority. *Operations Research*, 49(4):565–577, 2001.

[6] L. Ehlers, I. Hafalir, B. Yenmez, and M. Yildirim. School choice with controlled choice constraints: Hard bounds versus soft bounds. GSIA Working Papers 2012-E20, Carnegie Mellon University, Tepper School of Business, Nov. 2011.

[7] G. Haeringer and V. Iehlé. Two-sided matching with one-sided preferences. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, pages 353–353, 2014.

[8] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[9] J. Horton and K. Kilakos. Minimum edge dominating sets. *SIAM Journal on Discrete Mathematics*, 6(3):375–387, 1993.

[10] M. Jerrum. *Counting, Sampling and Integrating: Algorithms and Complexity (Lectures in Mathematics. ETH Zürich).* Birkhäuser Basel, 1 edition, Apr. 2003.

[11] T. Kavitha and M. Nasre. Popular matchings with variable job capacities. In *Proceedings of the 20th International Symposium on Algorithms and Computation*, ISAAC '09, pages 423–433, 2009.

[12] M. A. Satterthwaite and H. Sonnenschein. Strategy-proof allocation mechanisms at differentiable points. *The Review of Economic Studies*, 48(4):587–597, 1981.

[13] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23 – 37, 1974.

[14] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93 – 133, 1989.

[15] S. Toda. On the computational power of pp and (+)p. In *30th Annual Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[16] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201, 1979.